11. The apparatus of claim 10, wherein the means for using the program counter value to locate the component comprises means for converting the program counter value into a component name when the cache memory does not contain a component name at a location addressed by the program counter value and means for storing the component name in the cache memory.

12. The apparatus of claim 9, wherein the compiler controlling means comprises means for using the compiler to create, as part of the symbolic debugging information, at least one map which indicates a relation between the component object code and the component source code and means for associating the at least one map with the located component.

13. The apparatus of claim 9, wherein each computer program is constructed as a collection of components with dependencies between components, each component having an interface and an implementation and wherein all component dependencies are from component interfaces and wherein the compiler controlling means comprises means for controlling the compiler to recompile source code for the located component and all components which depend on the located component.

14. The apparatus of claim 9, wherein the compiler controlling means comprises means for updating symbolic debugging information which was originally created by compiling and linking all of the components.

15. The apparatus of claim 9, further comprising means for executing a browser program with the symbolic debugging information to present source code from a located component on the display when program execution halts in response to an exception generated by the program.

16. The apparatus of claim 15, wherein the executing means comprises means for applying a program thread which generated the exception as an input to the browser program.

17. A computer program product for use in a computer system having a memory, a display, a program counter with a value, a program consisting of a set of named components stored in a database in the memory, each component including an attribute indicating whether the component data is valid, source code for implementing the component, and object code for executing the component, a compiler, and a debugger for monitoring execution of the program to detect a program execution halt during debugging, a computer program product for dynamically generating symbolic debugging information comprising a computer usable medium having computer readable program code thereon, including:

(a) program code for using the program counter value to locate a component in the database when program execution halts during debugging;

(b) program code for checking the attribute of the located component to determine whether symbolic debugging information relating the object code to the source code is valid;

(c) program code for controlling the compiler to generate the symbolic debugging information by recompiling the source code of the located component when the symbolic debugging information is not valid;

(d) program code for associating valid symbolic debugging information with the located component; and

(e) program code for controlling the debugger to continuing debugging the program.

18. The computer program product of claim 17, wherein the program code for locating a component in the database comprises program code for using the program counter value to address a cache memory and program code for obtaining a program component name from the cache memory.

19. The computer program product of claim 18, wherein the program code for locating a component in the database comprises program code for converting the program counter value into a component name when the cache memory does not contain a component name at a location addressed by the program counter value and program code for storing the component name converted from the program counter value in the cache memory.

20. The computer program product of claim 17, wherein the program code for controlling the compiler comprises program code for using the compiler to create, as part of the symbolic debugging information, at least one map which indicates a relation between the component object code and the component source code and program code for associating the at least one map with the located component.

21. The computer program product of claim 17, wherein each computer program is constructed as a collection of components with dependencies between components, each component having an interface and an implementation and wherein all component dependencies are from component interfaces and wherein program code for controlling the compiler comprises program code for controlling the compiler to recompile source code for the located component and all components which depend on the located component.

22. The computer program product of claim 17, wherein program code for controlling the compiler comprises program code for updating symbolic debugging information which was originally created by compiling all of the components.

23. The computer program product of claim 17, further comprising program code executing a browser program with the symbolic debugging information generated by the compiler to present source code from a located component on the display when program execution halts in response to an exception generated by the program.

24. The computer program product method of claim 23, wherein the program code for executing the browser program comprises program code for applying a program thread which generated the exception as an input to the browser program.

* * * * *